Thermal Model Development for Ares I-X

Ruth M. Amundsen[1], Joe Del Corso[2]
NASA Langley Research Center, Hampton, VA 23666

Thermal analysis for the Ares I-X vehicle has involved extensive thermal model integration, since thermal models of vehicle elements came from several different NASA and industry organizations. Many valuable lessons were learned in terms of model integration and validation. Modeling practices such as submodel, analysis group and symbol naming were standardized to facilitate the later model integration. Upfront coordination of coordinate systems, timelines, units, symbols and case scenarios was very helpful in minimizing integration rework. A process for model integration was developed that included pre-integration runs and basic checks of both models, and a step-by-step process to efficiently integrate one model into another. Extensive use of model logic was used to create scenarios and timelines for avionics and air flow activation. Efficient methods of model restart between case scenarios were developed. Standardization of software version and even compiler version between organizations was found to be essential. An automated method for applying aeroheating to the full integrated vehicle model, including submodels developed by other organizations, was developed.

I. Introduction

Ares I-X is the upcoming test flight for the Crew Launch Vehicle (CLV), or Ares I. The Ares I-X vehicle is planned for launch in April 2009 from NASA Kennedy Space Center (KSC) in support of NASA's Constellation Program (CxP). The vehicle includes a standard active solid rocket booster (first stage) from the Shuttle Program. The upper stage and crew exploration vehicle portions of the vehicle are simulators, with a similar outer mold line (OML) to Ares I, but no active propulsion or crewed area in the upper stages. Thermal modeling has been performed for Ares I-X to show that the vehicle design is thermally robust, and that all components and materials survive the expected conditions. Since Ares I-X is in many ways a similar vehicle to Ares I, and also involves similar analysis to other NASA Exploration programs, the lessons learned on this early program can be used to improve thermal analysis on future CxP missions. Several integrated product teams (IPTs) come together to develop Ares I-X; these are First Stage (FS), Upper Stage Simulator (USS), Crew Module / Launch Abort System (CM/LAS), Avionics, and Roll Control System (RoCS).

II. Modeling Process

The Ares I-X thermal model has been developed in the thermal software Thermal Desktop®[1]. Most of the product teams (and all the NASA Centers) involved on Ares I-X were already using Thermal Desktop (TD), and it is the standard for use on Ares I, so it made a natural choice for the integrated vehicle model. The integrated vehicle model was built using input from several sources. The original CLV upper stage model, developed by NASA Marshall Space Flight Center (MSFC), was used as a base to start from; this model already had the KSC environment and a method for applying the ascent aeroheating built in. Several of the IPTs contributed their portions of the vehicle to the full vehicle thermal model. NASA Glenn Research Center (GRC) developed the USS model. Teledyne Brown Engineering developed the RoCS model. Lockheed Martin, leading the Avionics IPT, developed the model for the avionics elements mounted on the First Stage Avionics Module (FSAM) structure within the First Stage extra long (XL) segment (the FS segment above the four solid rocket segments). All other parts of the vehicle thermal model development, as well as the integration of the above models, was performed by the Systems Engineering and Integration (SE&I) IPT at NASA Langley Research Center (LaRC). The thermal model, shown in Figure 1, included the entire vehicle outer mold line, all structural masses necessary for thermal analysis, the pad structures, avionics, tanks, and propellant. Operation of fans and environmental control system

[1] Thermal Engineer, Structural and Thermal Systems Branch (D206), Mail Stop 431.
[2] Thermal Engineer, Structural and Thermal Systems Branch (D206), Mail Stop 431

(ECS) air conditioning were handled by portions of the model in FloCAD, a module of Thermal Desktop that handles fluid flow. Radiation between all vehicle and pad surfaces was included, as well as radiative and convective exchange with the outside environment, vehicle assembly building (VAB), and solar fluxes. The radiative modeling process for a vehicle on the surface of the planet is somewhat complex, and is described in an accompanying paper[2].

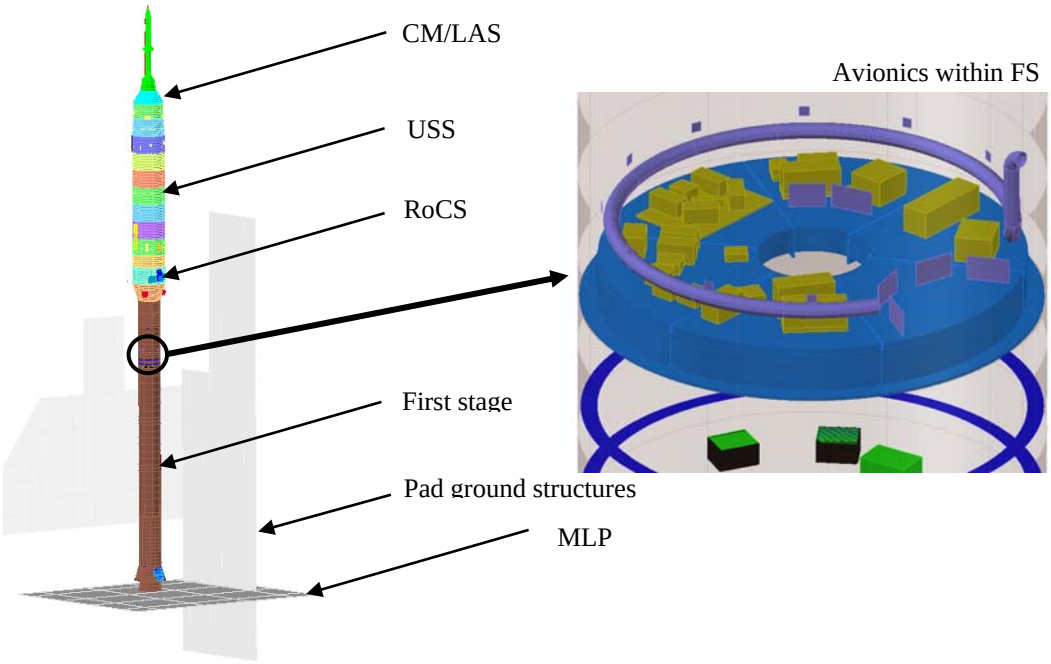

Figure 1. Ares I-X vehicle thermal model.

III. Lessons Learned

Over the last 20 months of thermal model development and analysis, much has been learned about the best methods to model a system of this magnitude and complexity. These lessons learned are described below by category: standardization of the model, efficient ways to integrate models, how to use model logic to facilitate case structure, how to do restarts on a combined thermal/fluid model, and the importance of software continuity between organizations.

A. Model Standardization

There are many items that can be defined in advance of the model development to make not only model development easier, but define common terminology and parameters across models and facilitate later integration. Many of these are simply good modeling practices in general. After importing multiple thermal models from three different organizations, it became obvious how much standardization of this kind helps when models are to be transferred.

Submodel naming. It greatly helps with later integration, as well as with the ability of an outsider to understand the model, if a consistent practice is used for naming submodels (including fluid submodels). Guidelines used on Ares I-X were to have the first three letters of the submodel indicate what part of the vehicle it was from (i.e., ABT for launch abort system, CEV for Crew Exploration Vehicle, USS for Upper Stage Simulator, FS_ for first stage, RCS for RoCS, GND for ground systems structures). This convention allowed the analyst to not only understand which part of the model any given item pertained to, but also to easily post every submodel having to do with one vehicle section (since they are listed alphabetically in the Model Browser).

Number of submodels. When a model is designed to be imported into another, it is beneficial to have guidelines on the number of submodels. It is very helpful to break a model into submodels to allow more efficient model development, viewing, and operation. However, if each model that is to be imported contains 50 submodels, the overall model will quickly become unwieldy. Conversely, if each imported model only contains one submodel, it

will be more difficult to understand the model structure and debug problems. In general, it is advantageous to establish guidelines on number of submodels for an imported model, either by overall number, or by the number of nodes that should be in a submodel, or by size or other criteria. For Ares I-X, each IPT was requested to supply a model with less than 20 submodels. The full vehicle model currently contains 53 submodels, which is a reasonable number to work with.

Number and naming of layers. The same ideas apply to layers. If each imported model contains 100 layers, the number of layers in the full model will quickly become unwieldy. Conversely, if each submodel contains only the default 'layer 0', understanding and presentation of the model is limited. The best practice is to name layers so they are understandable to others, discard ones unused, and keep the total number reasonable. If the model contains imported geometry that was used to 'snap' Thermal Desktop surfaces to, keep that in the model with an easily identifiable name. Filters can also be used with layers to facilitate display of various parts of the model. To facilitate viewing in solid shaded mode, use colors other than black for most layers.

Number and naming of radiation groups. The same ideas apply to radiation groups. The best practice is to keep the number limited, and make the name understandable, starting with the element in which it resides. Names of radiation groups that are to be used by more than one imported model must be consistent across models.

Units. Make sure there is an understanding on what units will be used, before each organization starts model development. This includes units used for fluid submodels.

Symbols. It is very helpful to define in advance some symbols that can be used in each model, and what units they are in, such that there is a common understanding between the model developers, and information and code can be shared. For example, on Ares I-X the standard practice for avionics was to have the dissipated power of the avionics box represented by the product of two symbols, one for the power value, and one a switch that turned the box on and off. In other words, for a box called ATVC, the power symbol would be pw_ATVC (in W), and the switch to turn the box on and off would be sw_ATVC. Power is then input in the heat load on the box as pw_ATVC* sw_ATVC, and the appropriate unit conversion used. This convention allowed the power to be changed easily in the model, or even changed temporarily for a what-if case, without delving into the model logic. It also allowed the model logic that turned boxes on and off to operate independently of setting the power values. Finally, it allowed dual use of switches – there could be several boxes that would be turned on with a single switch. As another example, the conductor for an item that changes between cases can be defined with a symbol: e.g., the hold-down-post conduction symbol can be set to go to zero on ascent, when that connection is broken.

It is best to have the initial values of symbols be something reasonable, even if those symbols will later be overwritten by model logic, so that on initial build the model will have reasonable parameters. If a symbol might later be referenced in model logic, it must be eight letters or less to allow it to be a register.

Symbol groups. In each imported model, keep the symbols in separate groups with understandable names that are unique from other symbol groups in the overall vehicle model. Then, if an imported model needs to update their symbols, these can be imported simply as a group, without risk of missing any or overwriting others. Also, this practice allows the symbols that correspond to certain submodels to be easily located.

Comments. For materials, the comment block should be used to give the reference that the data is from, to afford traceability. Similarly for any data or array used, the analyst should give the reference and date for traceability. For conductors, contactors, surfaces, etc, it is helpful to add text in the comment block to describe it, so that when the full model is assembled, the analyst can look down a list of contactors or conductors in the Model Browser and easily understand what each one represents.

Boxes. In general, if something has a box shape, model it using a box rather than 6 separate surfaces. It is best to give it a material with the appropriate density so the box mass is correct; this allows it to show up correctly when the total model mass is calculated. The node mass can be over-ridden with a symbol, but the box mass will not then show up summed with other masses.

Conductors/ties. The best practice is to consolidate conductors and ties where possible. In other words, if there are 8 surfaces with similar ties to the same fluid lump, or 8 boxes that all contact one surface, put them all together on a single form (with a descriptive comment), rather than on separate forms. Again, this practice allows the model to be understood and modified more easily.

Calculations. In expression boxes where a calculated number is entered, write out the entire expression rather than just entering a number (e.g., for a box using area * conductivity / length, write that in the comment box, and write 5 * 0.1 / 7 in the expression field rather than just 0.07). This practice allows others to understand where the value came from.

Case sets. Name case sets in the Case set manager fully to describe what they are.

Notes. Keep a list (the easiest place is in Utilities…Notes) of what basic changes have occurred with the model, and what nodes/conductors are the most critical. For the integrated model, keep a list of what version of each imported model is currently used.

Coordinate system. Define a coordinate system that will be used in all models to be imported. Each model can also use their own coordinate system of choice, but if at least one common coordinate system exists in all imported models, it will greatly facilitate model transfer and debugging. On Ares I-X, the common coordinate system between models was the vehicle coordinate system from the official Outer Mold Line (OML) document.

B. Model Integration

Many important lessons were learned to smooth the process of model import and integration.

1. Pre-model development coordination

The first was to use all the model standardization tips above. In particular, if there are general symbols that should be used by all modelers, define those in a template model, as well as symbol groups, radiation groups and any common materials. Common symbols for Ares I-X included STRTIME and RUNTIME, variables that were used for the start time and run time of each case. Others were OFFSETL and TIME2L for the launch time using a 24 hour clock, and the time left until launch (in hours). This practice allowed each modeler to write logic that utilized a common time framework for each case. Environmental parameters SKY_TMP, AIR_TMP and GND_TMP were defined for the sky (radiative sink), air and ground temperatures. These symbols eventually referenced arrays for the diurnal swing of those temperatures over a daily period. The Ares I-X model included many scenarios and several cases, which were in general defined by model logic described in the next section. To make sure that all model developers used the same symbols to define the case scenario, two symbols were pre-defined. One was loc_def, which had different values depending on the case location (0 for the VAB case, 1 for rollout, 2 for on-pad, etc). The second was casedef, which was 0 for cold case, 1 for hot case. Other common symbol standardization tips are to start all thicknesses with 'th_', contact conductances with 'cc_', and define other similar nomenclature guidelines as necessary. In order to avoid duplication of common materials, it is helpful if the template drawing has many of the common materials predefined. When modelers add new materials that may be used by others, they can distribute them, so that the material properties files remain identical and no material duplication will occur after models are integrated.

Coordination between all parties developing models, before models are built, is critical to ensure that the guidelines above are followed, and that standard guidelines are set. Also important to discuss is the method that will be used for on-planet and ascent/descent solar modeling.

Guidelines for what should be delivered with a model will be beneficial to ensure that enough material is delivered with the model to assist in model integration. When the model is delivered, it should include necessary material property files as well as any external files needed to run. It should also include a document that describes at least the submodels, basic symbols, cases, model logic and results. At least one results case (either .sav file or a transient plot and gradient image) should be supplied to compare to.

2. Model checking

When models are submitted to be imported into an overall model, it is important to check them thoroughly before importing them and potentially damaging a working vehicle model. When a model is transferred, the first task is to open it and define the new location(s) of the optical and material property files that were transferred with it. The following items were found to be useful to check before model import:

- View model in Model browser by Thermal properties, by Optical properties, and by Radiation groups. Make sure none are default. Make sure that everything makes sense, and in particular that the radiation groups have identical names as any that are to be matched in the integrated model.
- Post by radiation groups to make sure right surfaces/solids are in each.
- Run a mass check (Thermal -> Model Checks -> Calculate Mass) and make sure the model mass is correct.
- Output SINDA Con/Cap data to make sure there are no errors.
- Look for duplicate nodes (Thermal -> Model Checks -> List Duplicate Nodes).
- View active sides to verify their definition is correct.
- Check units for both thermal model and fluid model.
- Run the submitted model and check both an image of the results and a plot of the transient results against the results supplied by the model developer (results can differ due to differences in software version, compiler versions, or errors in external file locations).

3. Model import process

Currently, the process of importing one Thermal Desktop model into another involves several steps. It is hoped that at some point in the future this process will be smoother. However, for Ares I-X, the following process was followed with each imported model.

1. Ensure the integrated model runs before import of submitted model. Save a results file to check against.
2. Export from the submitted model: symbols, correspondence data, and property aliases (Logic Manager objects, orbits and case sets are optional).
3. In submitted TD model, display everything and all desired submodels. Be careful to display all layers, submodels, and nodes that are to be imported. Contactors that are attached to something not exported will not come across. A handy option, if a contactor exists that is attached to something not to be exported (i.e., structure that already exists in the integrated model) is to temporarily alter it so that it ties to one of the exported surfaces. Then after the import it can be corrected back to the original location.
4. Set correct coordinate system and units for transfer.
5. Export wblock of desired items. (Type 'wblock', click source=objects, and select all items to be exported/imported. Be careful that the units drop-down names the correct units). Pick a logical file name (usually including '_block' to denote the block export).
6. Open integrated model and set to correct coordinate system and units. If importing fluid submodels, set the FLUINT output units, and add any necessary fluid model constituent fluids.
7. Import the symbols and property aliases exported in step 2, and materials from the submitted materials files, using import buttons in those forms. (Logic Manager objects, orbits and case sets are optional). As mentioned above, to avoid duplication of common materials, it is helpful to have all models started from a template which includes the same common materials. Then, only the materials unique to the submitted model must be imported.
8. Do Insert -> Block and pick the block drawing file just created. Make sure the Explode box is checked. Do not select a location and rotation – if it is in the correct co-ordinate system, it will come across in the same place.
9. After the import, to have those submodels appear in all the drop-down menus, do Thermal ->SINDA Submodels and select Scan DB.
10. Add any new radiation groups to the model by doing Thermal ->Radiation Groups and select Scan DB.
11. Create any contactors/conductors/ties that may be needed to connect the imported model into the base model. Correct any contactors or conductors that were artificially set to contact within the imported block.
12. Import correspondence data.
13. Check total model mass.
14. Add any new radiation cases to all applicable cases in the case set manager (use multiple edit mode to change several cases at a time).
15. Add any necessary registers to case sets.
16. View model in Model Browser by Thermo Props and Optical Props. If materials are duplicated by similar entries, determine if the duplicate material has the same properties and thus can be eliminated and its surfaces transferred to consolidate into one material entry.
17. View model by symbols to make sure usage is consistent and no symbols have been duplicated by mis-spellings.
18. Run the same case run in step 1 to check that results have not changed in any inappropriate manner. Test the performance of the submitted model by running the same case as the results submitted, and check that results are compatible. Several of the output subroutines can be used to evaluate the behavior of the imported submodels relative to the integrated model, such as SUBMAP to show thermal maps between each submodel.

Although this process involves many steps, it is actually fairly rapid to accomplish, and gives repeatable results.

C. Model Logic

The Ares I-X vehicle model, besides incorporating many different submodels for different physical systems, also incorporates many event scenarios. Logic is used within the model to select cold case versus hot case for all environment temperatures and solar flux cases. Logic is also used to select if the ECS system is on or off, if cooling

fans are on or off, is there will be a hold before launch, and if the case is for a post-launch abort case. In the Ares I-X model, 16 logic blocks within the Logic Manager are used to define the necessary scenarios. The advantage to handling this within the logic manager as opposed to placing them within the logic for a single case within the Case Set Manager is that the logic can be developed to apply to all cases, and can be simply altered in one location as necessary. In each case, registers are set to define the parameters for that case, which change the effect of the model logic block.

The array interpolation function is used to set up arrays for hot and cold case air temperatures versus time, and radiative sky temperatures versus time. The user FORTRAN code option is used in the remaining blocks. In Operations block pre-build are placed all logic lines that need to be done only once. An example is the lines that define the array to be used for external temperature based on the casedef register. If the Loc_def value indicates a VAB run, then the external temperature used is for the VAB. Logic that depends on calling nodes from submodels is placed in the Operations Post-build block, such as setting temperatures for the solid rocket igniter heater and forward dome above the propellant, as well as the aft skirt air, based on the case. A Variables 0 block includes all code that must be updated with time, such as calls to the correct time point in arrays for the time-dependent external temperatures. Also in this block is a segment that increases the run time if this is a hold case.

The four most complex logic blocks are those that control the time sequencing of avionics and ECS in the FS and USS in the time leading up to launch. Using the TIME2L register to define the remaining time before launch allows the pre-launch sequencing to be independent of time of day, and thus allows many different cases for various launch times to be easily run. A USS submodel Variables 0 block implements the entire time sequence for sequential turn on and off of all USS avionics boxes and cooling fans, based on the case definition, time to launch, and whether this is a hold or abort case. A similar block exists for the FS avionics, which controls the sequencing of all FS avionics power. Flogic 0 blocks are used for the air flow control in FS and USS, to turn off ECS in the planned sequence prior to launch, and re-activating it in the case of an abort.

Additional blocks are used for detailed printout in certain cases, such as performing heat maps while ECS is running to determine the heat load on the air exchanger. The output time step is controlled in the on-pad case leading up to T-0, so that the results can be viewed on a finer scale in the minutes before launch.

These logic blocks can be easily exchanged with other modelers, using the import/export functions, which saves modeling time as long as the registers to be used have been set up correctly in the beginning.

D. Case Integration: Restart Method

When running a complex vehicle model, there are usually many cases to be taken into account. These are in general run by using different cases in the Case Set Manager. For Ares I-X, there are hot and cold cases in the VAB, then rollout, on-pad, ascent and descent. Each of these cases may have several different hot and cold cases, with differing times of year, wind conditions, etc. Some cases may have differing scenarios afterward (the on-pad case may lead to launch, a hold, or an abort). In order to have the most flexibility, it is usually easiest to run each scenario as a different case, and then restart the run for one case from the applicable time of the prior case. In addition, certain cases must be run separately, as with the difference between rollout and on-pad, when a different radiative environment is present (i.e., the pad structures).

In general, restarts had been done by using the Advanced tab in the Case Set Information, and setting a save file from a previous run to start from. However, this process currently only carries temperatures for thermal nodes over between cases, and does not carry over temperatures for fluid lumps. For Ares I-X, some of the air lumps are massive (hundreds of pounds of air), and thus are important to carry across correctly between cases, to have the correct starting conditions. To do this, the ‘Set Initial Temperatures’ check box in version 5.1 of Thermal Desktop must be cleared, and the restart done using logic calls. The restart is accomplished by writing more information to the save file (including fluid lump temperatures), and then restarting using the RESTAR command. The save file for the initial case must be output using at least the ‘Temperatures’ and ‘Lump Info’ checkboxes (C&R recommends also Flowrates and Tie Info, although this is not always possible between cases). Then, in the next case, the RSI file (in Case Set SINDA Options) is defined as the save file from the previous run. In Operations Data, RESTAR is called using the record number from the desired output time of the save file (this is printed in the output file from the previous run). This command assigns initial temperatures to both the thermal nodes and fluid lumps from the previous at that defined time. For Ares I-X, the FORCER command is then used to make sure that all registers are at their defined values for the new case, and TIME0 and TIMEND are refined using the current STRTIME and RUNTIME registers. (This is normally only necessary if registers were output to the restart save file).

In certain cases, because the storage of fluid lump variables is complex for tanks, it has been found that this method will not be successful with only T and L in the save file, and that all lump parameters must be stored. It is anticipated that future developments in Thermal Desktop and the FloCAD module will make this process smoother.

A routine has been developed in a beta release of TD that allows restart from fluid lump temperatures, and this is anticipated to be available in the next release.

E. Software Standardization

One unexpected lesson learned from integrating models from various organizations was that the specific Fortran compiler used can have a big impact. One organization that delivered a model was using the Intel Fortran compiler, and the integrated model was being run with the built-in compilers within Thermal Desktop (Lahey compiler). When the submitted model was run with the built-in compiler version, it exhibited much different behavior than designed, and in some cases would not run. This incident highlighted the need for standardization of not only the version of Thermal Desktop and AutoCAD used (which had been done in advance), but also the compiler version used. If this is standardized before model development or submittal, the model transfer process will avoid these potential roadblocks.

IV. Automated Aeroheating Application

A. Body point mapper (BPMapper)

For the ascent phase, aeroheating was applied to the exterior of the Ares I-X vehicle. The aeroheating was developed by NASA MSFC using the MINIVER software, which calculates the heating at different body points over the vehicle (rather than using a full vehicle mesh). This process is much faster than CFD, and produces a full ascent trajectory of aeroheating, with time steps of less than a second, so that all changes in heating during the ascent are captured. However, since it uses a relatively coarse selection of body points, that heating must be carefully mapped onto the vehicle model. Also, the skin temperature of the vehicle must be taken into account in calculating the heat flux at a certain point and time.

The aeroheating body point files from MSFC gave at each time point in the trajectory, the fluid enthalpy, wall enthalpy, and heat transfer coefficient for three different skin temperatures of the vehicle (0, 760 and 2000°F). There was software within the thermal model that computed the heat flux based on the skin temperature of the node, by interpolating between the available skin temperatures. The more difficult task was to assign body points to locations on the model, since there are about 2000 body points, and more than 7000 TD nodes. This task was accomplished by software called BPMapper, originally developed at NASA GRC and modified extensively by NASA LaRC. BPMapper takes the locations of each TD node on the surface of the vehicle, and maps it to the closest body point from the heating data. The software allows for custom mapping, to 'lock' sets of nodes to certain body points or ranges of body points, if the analyst wants to pre-define them and does not want the program to do the mapping.

Since Ares I-X aeroheating is highest on the protuberances, it is usually critical to make sure that heating for protuberances is applied only to protuberances in the model, and clean skin (acreage) heating is applied only to clean skin surfaces. To ensure that protuberance and clean skin heating remain divided, the thermal submodels are broken into 'clean skin' and protuberance submodels. The aeroheating body points are numbered such that those ending in numbers above 19 are protuberances. Thus, the BPMapper code is able to distinguish between protuberance and clean skin heating, and apply them correctly.

The complete process and files required for running BPMapper are defined in the readme file for the code, but an overview is provided here. The user can define whether or not they need to apply a coordinate transformation: i.e., whether the body points and Thermal Desktop nodes are in the same coordinate system. The process for using BPMapper is to first write out all node locations from Thermal Desktop. The body point locations are also written out to a file. The thermal submodel names for clean skin and protuberances are defined in separate text files. Then the BPMapper code is run to perform the mapping, and generate arrays of the body points to be used for heating in the Thermal Desktop run. For verification purposes, the code writes out files to let the user know how far each mapped TD node is from the body point it was mapped to. BPMapper also writes out a file in Thermal Desktop's 'text transient' file format, which shows the body point used graphically on the vehicle; a sample for the nose body points is shown in Figure 2. When the Thermal Desktop thermal case is run, the code associated with the aeroheating also outputs a file in text transient file format, which allows the user to graphically plot the heating applied to the vehicle at each time point (Figure 3). This graphical verification helps identify incorrectly mapped nodes as well as places where the aeroheating files are in error. For Ares I-X, graphical verification was used to find places where engine plumes that existed in the aeroheating data were incorrect (for engines that were only simulators on Ares I-X).

One of the main advantages of this automated method was that, when a new model was imported, or an external portion of the model was remeshed, the body points could be re-mapped within minutes. This method avoided the

tedious manual pairing of body points with thermal nodes, and eliminated the rework associated with remeshing. The graphical verification was critical to avoid errors and highlight issues.

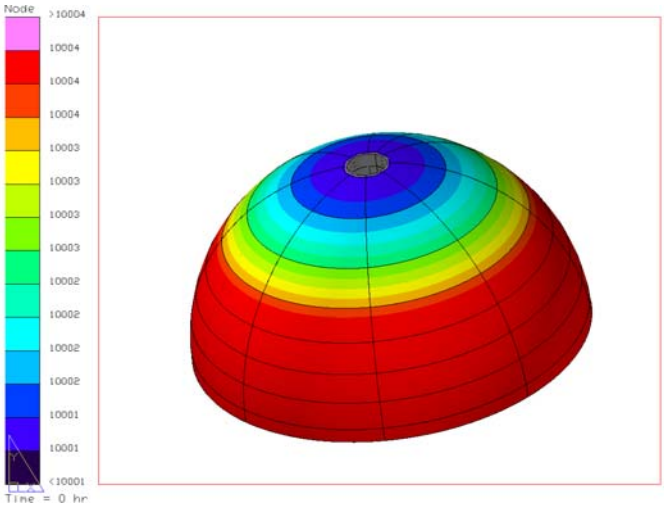

Figure 2. Body point number mapping on Thermal Desktop model.

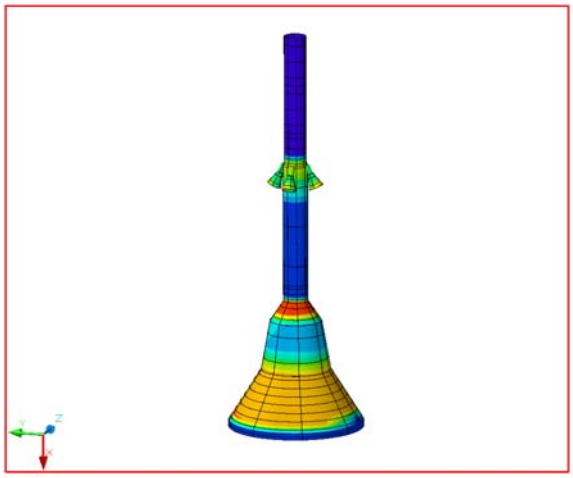

Figure 3. Example aeroheating flux from text transient file, on launch abort portion of vehicle.

B. Boundary condition mapper

Cullimore and Ring recently added the Boundary Condition Mapper (BCM) to Thermal Desktop, in version 5.2. The boundary condition mapper functionally allows the user to take a BCM formatted input text file and map heat flux to Thermal Desktop models. C&R's unique BCM text input file is formatted such that it defines nodal positions, polygon connectivity (triangles defined by nodes), temperature dependent heat flux, and units.

Heat flux loads for portions of the Ares I-X model were generated using Computational Fluid Dynamics (CFD) and output in 3D Tecplot geometry format. The Tecplot result files were converted to BCM format input files using Map2CFD. Map2CFD, developed at NASA LaRC and written in Perl in order to leverage the language's file parsing capability, reads in the Tecplot node, heat flux, and triangulated polygon information from several 3D Tecplot CFD results files generated at different wall temperatures. Map2CFD currently allows the user, via command line inputs, to scale and/or invert the input heat flux, and will combine multiple Tecplot geometries into a single geometry, and then write to both BCM and Tecplot format. The ability to export to Tecplot was implemented to give the user the ability to verify that all data was correctly generated. Map2CFD requires that the user generate a mapping file which allows Map2CFD to locate results files and order them correctly. Additional information can be found in the Map2CFD documentation.

These heat flux loads can then be applied to the model using the Boundary Condition Mapper, and used in a thermal run. This again allows a much simpler and more accurate application of aeroheating, and avoids the manual mapping of past methods.

V. Conclusion

Ares I-X vehicle thermal analysis has provided many valuable lessons for the process of developing an entire vehicle model from smaller models of vehicle segments. Guidelines were developed for the standardization of vehicle segment thermal submodels which substantially lowered the time necessary to integrate them with the vehicle model. Guidelines and recommendations for pre-model development coordination were defined. A process was developed for import of submodels which avoided time delays and errors on import. Extensive use was made of symbols and model logic to simplify case development and modification. Two automated methods for applying aeroheating to the entire vehicle model were developed. It is hoped that some of these lessons can be beneficial for future missions, which will have a substantial level of model integration for vehicle-level models.

Acronyms

CFD	Computational Fluid Dynamics
CLV	Crew Launch Vehicle
CM/LAS	Crew Module / Launch Abort System
CxP	Constellation Program
ECS	Environmental control system
FS	First Stage
FSAM	First Stage Avionics Module
GRC	Glenn Research Center
IPT	Integrated product team
KSC	Kennedy Space Center
LaRC	Langley Research Center
MSFC	Marshall Space Flight Center
OML	Outer mold line
RoCS	Roll Control System
SE&I	Systems Engineering and Integration
TD	Thermal Desktop
USS	Upper Stage Simulator
VAB	Vehicle Assembly Building
XL	Xtra long

Acknowledgments

The methods for applying aeroheating to this model, as well as initial model structure, were supplied by Mark Wall of MSFC, from the CLV US thermal model. The USS portion of the model was originally built by Josh Giegel, and modified by Marcus Studmire, Jim Yuko, Bob Christie and Jim Myers (NASA GRC). RoCS submodels were supplied by Preston Beatty (TBE). FSAM submodels were supplied by Gary Holmstead (LMA). The expertise of MSFC personnel in supplying aeroheating (Mark D'Agostino, Craig Schmitz, Jason Mishtawy, and Colin Brooks) is gratefully acknowledged. The technical support from the team at Cullimore & Ring was outstanding. The assistance of Tory Scola at NASA LaRC in development of the BPMapper code is gratefully acknowledged.

References

[1] Thermal Desktop User Manual, Cullimore and Ring Technologies, Inc., Version 5.1, October 2007.
[2] Gasbarre, J. F, Amundsen, R. M., and Scola, S., "Ground Plane and Near-Surface Thermal Analysis for NASA's Constellation Programs," *Thermal and Fluids Analysis Workshop, TFAWS 2008*, San Jose, California, August 18-22, 2008.